Lab 2 Report: SDN Simulation

** Please **fill in the report** and submit the **pdf** to NYU Brightspace*

Name: Yujie Yu, Junda Ai ID: yy3913, ja4426 Date: 10/06/2022

1. Objectives

- Fully understand the operation of Openflow and observe the operations
- Master the simulation tool mininet

2. Equipment Needs

- Computers
- Internet

3. Experiments

3.1 Basics

1. Install Ubuntu on your computer, you can install it on a VM using either VMWare player or VirtualBox:
   VMWare:
   https://my.vmware.com/web/vmware/free#desktop_end_user_computing/vmware_player/6_0
   VirtualBox:
   https://www.virtualbox.org/wiki/Downloads
   Ubuntu 16 or 18 is recommended! Ubuntu 20 might have some issues with python 2 and mininet!!
2. Follow the instructions to set up Mininet on your Ubuntu VM:
   http://mininet.org/download/
3. Perform basic simulations following these steps:
   http://mininet.org/walkthrough

3.2 Openflow

Part 1. Observe Openflow control messages:

1. Launch Mininet with default controller (**No "--controller remote" !**):

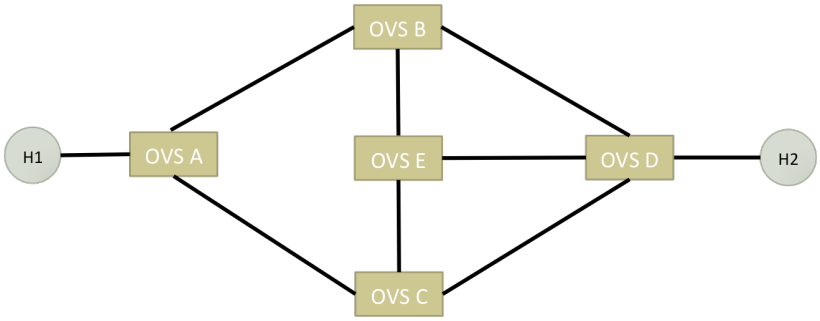
   $ sudo mn --custom ~/mininet/custom/topo-2sw-2host.py --topo mytopo

2. Install and open Wireshark in **Ubuntu**. Start listening to all the interfaces.

3. In Mininet:

mininet> pingall

You will see successful connections. Stop the Wireshark and find Openflow control flows. Fill in the table 4(a)



Part 2. Manually install flow entries in the OVS's

1. Use Mininet to create the topology above, where A, B, C, D, and E are all Openflow switches. (Remember to add **"--controller remote"** to disable default controller)
2. Enforce the following policies so that,
   a. Traffic from H1 ➔ H2
      i. HTTP traffic with d_port=80 follows path: A-B-D
      ii. other traffic follows path: A-C-E-D
   b. Traffic from H2 ➔ H1
      i. HTTP traffic with s_port=80, follow path: D-C-A
      ii. other traffic, follow path: D-B-E-C-A

   (i.e., you need to define a good match-action for the switches to achieve these.)

   c. verify your policies by,
      i. generating corresponding traffic (**using network debugging tools**)
      ii. capturing packets with Wireshark (listen to some switches' interfaces to see if the packets pass by)

   You can use OVS-OFCTL to manually install rules on switches (preferred method), or you can install a simple controller using RYU/POX/NOX/Beacon (Not recommended for lab 2 you'll do it in lab 3).

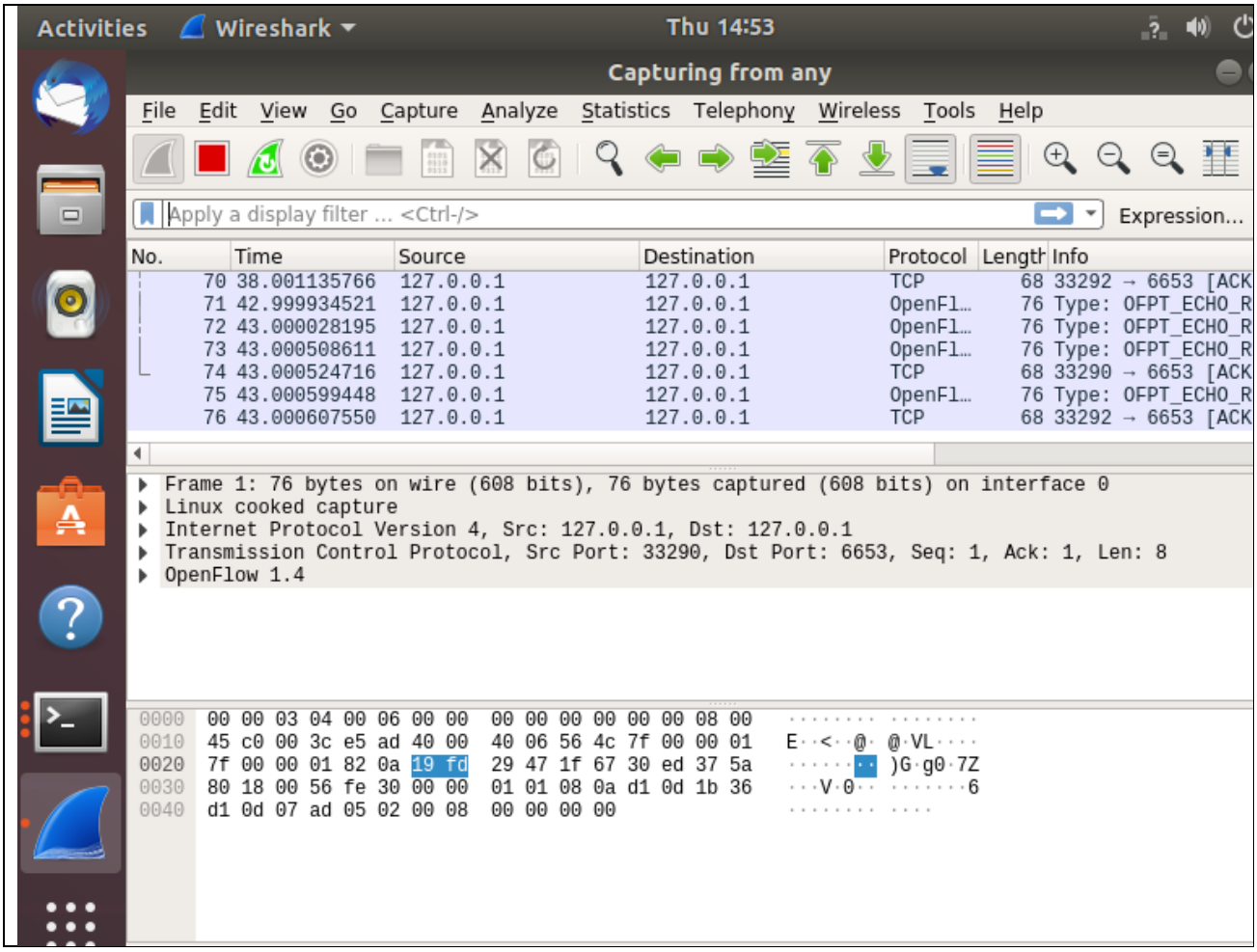
Part 3. Write a program that automatically creates a fat-tree with a given size N

1. Using mininet, create a 2-stage Fat Tree network using N-port switches, where N is a variable that should be an even number. (You don't need to check the connectivity, just create the topology.)

   (Set N as a global variable in the topology (python) file)

4. Reports

(a) A screenshot of OpenFlow control messages you captured with WireShark.



Commands	Meaning of the commands
ovs-ofctl show s1	Open virtual switch Open flow control Prints information on the switch especially regarding the flow tables and ports

	yuji@yuji-VirtualBox:~$ sudo ovs-ofctl show s1 [sudo] password for yuji: OFPT_FEATURES_REPLY (xid=0x2): dpid:0000000000000001 n_tables:254, n_buffers:0 capabilities: FLOW_STATS TABLE_STATS PORT_STATS QUEUE_STATS ARP_MATCH_IP actions: output enqueue set_vlan_vid set_vlan_pcp strip_vlan mod_dl_src mod_dl_ dst mod_nw_src mod_nw_dst mod_nw_tos mod_tp_src mod_tp_dst 1(s1-eth1): addr:46:0d:ff:43:35:1c config: 0 state: 0 current: 10GB-FD COPPER speed: 10000 Mbps now, 0 Mbps max 2(s1-eth2): addr:26:40:49:b2:70:0e config: 0 state: 0 current: 10GB-FD COPPER speed: 10000 Mbps now, 0 Mbps max 3(s1-eth3): addr:d6:0b:ae:87:4c:2e config: 0 state: 0 current: 10GB-FD COPPER speed: 10000 Mbps now, 0 Mbps max LOCAL(s1): addr:1a:bf:4c:34:ce:46 config: PORT_DOWN state: LINK_DOWN speed: 0 Mbps now, 0 Mbps max OFPT_GET_CONFIG_REPLY (xid=0x4): frags=normal miss_send_len=0
ovs-ofctl dump-flows s1	Open virtual switch; Open flow control Dump-flows: prints all the flow entries so matches and actions for the given switch yuji@yuji-VirtualBox:~$ sudo ovs-ofctl dump-flows s1 cookie=0x0, duration=11.894s, table=0, n_packets=0, n_bytes=0, in_port="s1-eth 1" actions=output:"s1-eth2"
ovs-ofctl add-flow s1 in_port=1,actions=output:2	Open virtual switch; Open flow control Add-flow : adds entry to the s1's switch tables to port 1 Actions: list of actions to do with a packet when the flow entry is matched Output: specifies what type of output yuji@yuji-VirtualBox:~$ sudo ovs-ofctl add-flow s1 in_port=1,actions=output:2 yuji@yuji-VirtualBox:~$ sudo ovs-ofctl show s1 OFPT_FEATURES_REPLY (xid=0x2): dpid:0000000000000001 n_tables:254, n_buffers:0 capabilities: FLOW_STATS TABLE_STATS PORT_STATS QUEUE_STATS ARP_MATCH_IP actions: output enqueue set_vlan_vid set_vlan_pcp strip_vlan mod_dl_src mod_dl_ dst mod_nw_src mod_nw_dst mod_nw_tos mod_tp_src mod_tp_dst 1(s1-eth1): addr:46:0d:ff:43:35:1c config: 0 state: 0 current: 10GB-FD COPPER speed: 10000 Mbps now, 0 Mbps max 2(s1-eth2): addr:26:40:49:b2:70:0e config: 0 state: 0 current: 10GB-FD COPPER speed: 10000 Mbps now, 0 Mbps max 3(s1-eth3): addr:d6:0b:ae:87:4c:2e config: 0 state: 0 current: 10GB-FD COPPER speed: 10000 Mbps now, 0 Mbps max LOCAL(s1): addr:1a:bf:4c:34:ce:46 config: PORT_DOWN state: LINK_DOWN speed: 0 Mbps now, 0 Mbps max OFPT_GET_CONFIG_REPLY (xid=0x4): frags=normal miss_send_len=0 yuji@yuji-VirtualBox:~$ yuji@yuji-VirtualBox:~$ sudo ovs-ofctl dump-flows s1
ovs-ofctl add-flow s1 priority=500,in_port=1,dl_type=0x0800,nw_proto=6, actions=output:2	(what are the matches and what is the action?) Open virtual switch; Open flow control add-flow: adds entry to the s1's switch tables Priority a value between 0 – 65535; and the

higher values will match before a lower one;
in_port = 1: where is the packet coming from
dl_type = 0x0800: matches ipv4 source/destination IP address

nw_proto = 6: protocol number between 0-255
actions: list of actions to do with the packet, when the flow entry is matched
Output: specifies what type of output

```
yuji@yuji-VirtualBox:~$ sudo ovs-ofctl add-flow s1 priority=500,in_port=1,dl_ty
pe=0x8000,nw_proto=6,actions=output:2
2022-10-12T00:09:27Z|00001|ofp_match|INFO|normalization changed ofp_match, deta
ils:
2022-10-12T00:09:27Z|00002|ofp_match|INFO| pre: in_port=1,dl_type=0x8000,nw_pro
to=6
2022-10-12T00:09:27Z|00003|ofp_match|INFO|post: in_port=1,dl_type=0x8000
```

Question: What is “priority”? Why do we need “Priority”? What is the default priority if the priority is not given?
Priority: allowing the certain packets to be processed earlier than other.
We use it to control what we consider sensitive behavior from the heavy flow of traffic of packages.
Defaults to 0 if unspecified.

Question: What is the default priority if the priority is not given?
Defaults to 0 if unspecified.

(b)

(c) Output of mininet “net” command for both topologies (part 2 & part 3). (you can use any N for Fat-tree, ex: 4, 6, 8)

2 Host 5 Switch Network

```
mininet> net
h1 h1-eth1:s1-eth1
h2 h2-eth2:s4-eth2
s1 lo:  s1-eth1:h1-eth1 s1-eth2:s2-eth1 s1-eth3:s3-eth1
s2 lo:  s2-eth1:s1-eth2 s2-eth2:s4-eth1 s2-eth3:s5-eth2
s3 lo:  s3-eth1:s1-eth3 s3-eth2:s5-eth3 s3-eth3:s4-eth4
s4 lo:  s4-eth1:s2-eth2 s4-eth2:h2-eth2 s4-eth3:s5-eth1 s4-eth4:s3-eth3
s5 lo:  s5-eth1:s4-eth3 s5-eth2:s2-eth3 s5-eth3:s3-eth2
c0
```

Fat Tree

```
mininet> net
h0
h1
h2
h3
h4 h4-eth0:s5-eth3 h4-eth1:s5-eth4
h5 h5-eth0:s4-eth3 h5-eth1:s4-eth4
h6 h6-eth0:s3-eth3 h6-eth1:s3-eth4
h7 h7-eth0:s2-eth3 h7-eth1:s2-eth4
s0 lo:  s0-eth1:s2-eth1 s0-eth2:s3-eth1 s0-eth3:s4-eth1 s0-eth4:s5-eth1
s1 lo:  s1-eth1:s2-eth2 s1-eth2:s3-eth2 s1-eth3:s4-eth2 s1-eth4:s5-eth2
s2 lo:  s2-eth1:s0-eth1 s2-eth2:s1-eth1 s2-eth3:h7-eth0 s2-eth4:h7-eth1
s3 lo:  s3-eth1:s0-eth2 s3-eth2:s1-eth2 s3-eth3:h6-eth0 s3-eth4:h6-eth1
s4 lo:  s4-eth1:s0-eth3 s4-eth2:s1-eth3 s4-eth3:h5-eth0 s4-eth4:h5-eth1
s5 lo:  s5-eth1:s0-eth4 s5-eth2:s1-eth4 s5-eth3:h4-eth0 s5-eth4:h4-eth1
c0
```

(d) The command you use to start your Mininet topology and the screen shot of the Mininet output while creating the networks.

Command you use to start your Mininet topology:
(Remember to add **"--controller remote"** to disable default controller; otherwise, you'll get zero for the following tasks.)

2 Host 5 Switch Network

```
yuji@yuji-VirtualBox:~/ctest$ sudo mn --custom topo.py --topo mytopo --controll
er remote
*** Creating network
*** Adding controller
Unable to contact the remote controller at 127.0.0.1:6653
Unable to contact the remote controller at 127.0.0.1:6633
Setting remote controller to 127.0.0.1:6653
*** Adding hosts:
h1 h2
*** Adding switches:
s1 s2 s3 s4 s5
*** Adding links:
(h1, s1) (h2, s4) (s1, s2) (s1, s3) (s2, s4) (s2, s5) (s3, s4) (s3, s5) (s4, s5
)
*** Configuring hosts
h1 h2
*** Starting controller
c0
*** Starting 5 switches
s1 s2 s3 s4 s5 ...
*** Starting CLI:
```

Fat Tree

```
yuji@yuji-VirtualBox:~/ctest$ sudo mn --custom fattree.py --topo mytopo --contr
oller remote
*** Creating network
*** Adding controller
Unable to contact the remote controller at 127.0.0.1:6653
Unable to contact the remote controller at 127.0.0.1:6633
Setting remote controller to 127.0.0.1:6653
*** Adding hosts:
h0 h1 h2 h3 h4 h5 h6 h7
*** Adding switches:
s0 s1 s2 s3 s4 s5
*** Adding links:
(s0, s2) (s0, s3) (s0, s4) (s0, s5) (s1, s2) (s1, s3) (s1, s4) (s1, s5) (s2, h7
) (s2, h7) (s3, h6) (s3, h6) (s4, h5) (s4, h5) (s5, h4) (s5, h4)
*** Configuring hosts
h0 *** defaultIntf: warning: h0 has no interfaces
h1 *** defaultIntf: warning: h1 has no interfaces
h2 *** defaultIntf: warning: h2 has no interfaces
h3 *** defaultIntf: warning: h3 has no interfaces
h4 h5 h6 h7
*** Starting controller
c0
*** Starting 6 switches
s0 s1 s2 s3 s4 s5 ...
*** Starting CLI:
```

(e) Briefly explain how you produce different traffic to verify whether the rules installed function correctly. (Hint: Use the network debugging tool in lab 1)

No traffic occurs

(f) With the produced traffic, show the screenshots of Wireshark capture on different links (switch with interface) to verify the paths taken by different traffic are correct.

Wireshark not check for that

(g) OVS-OFCTL commands used to install the rules on switches (part 2). (If you use a controller, upload your controller program)

TCP H1 => H2

```
yuji@yuji-VirtualBox:~$ sudoovs-ofctl add-flow s1 priority=500,in_port=1,dl_typ
e=0x8000,nw_proto=6,actions=output:2
sudoovs-ofctl: command not found
yuji@yuji-VirtualBox:~$ sudo ovs-ofctl add-flow s1 priority=500,in_port=1,dl_ty
pe=0x8000,nw_proto=6,actions=output:2
[sudo] password for yuji:
2022-10-12T00:52:20Z|00001|ofp_match|INFO|normalization changed ofp_match, deta
ils:
2022-10-12T00:52:20Z|00002|ofp_match|INFO| pre: in_port=1,dl_type=0x8000,nw_pro
to=6
2022-10-12T00:52:20Z|00003|ofp_match|INFO|post: in_port=1,dl_type=0x8000
yuji@yuji-VirtualBox:~$ sudo ovs-ofctl add-flow s3 priority=500,in_port=1,dl_ty
pe=0x8000,nw_proto=6,actions=output:2
2022-10-12T00:52:47Z|00001|ofp_match|INFO|normalization changed ofp_match, deta
ils:
2022-10-12T00:52:47Z|00002|ofp_match|INFO| pre: in_port=1,dl_type=0x8000,nw_pro
to=6
2022-10-12T00:52:47Z|00003|ofp_match|INFO|post: in_port=1,dl_type=0x8000
yuji@yuji-VirtualBox:~$ sudo ovs-ofctl add-flow s4 priority=500,in_port=1,dl_ty
pe=0x8000,nw_proto=6,actions=output:2
2022-10-12T00:53:00Z|00001|ofp_match|INFO|normalization changed ofp_match, deta
ils:
2022-10-12T00:53:00Z|00002|ofp_match|INFO| pre: in_port=1,dl_type=0x8000,nw_pro
to=6
2022-10-12T00:53:00Z|00003|ofp_match|INFO|post: in_port=1,dl_type=0x8000
```

Other H1 => H2

```
yuji@yuji-VirtualBox:~$ sudo ovs-ofctl add-flow s4 in_port=1,actions=output:1
yuji@yuji-VirtualBox:~$ sudo ovs-ofctl add-flow s2 in_port=1,actions=output:3
yuji@yuji-VirtualBox:~$ sudo ovs-ofctl add-flow s5 in_port=1,actions=output:3
yuji@yuji-VirtualBox:~$ sudo ovs-ofctl add-flow s3 in_port=1,actions=output:1
yuji@yuji-VirtualBox:~$ sudo ovs-ofctl add-flow s1 in_port=1,actions=output:1
```

TCP H2=>H1

```
yuji@yuji-VirtualBox:~$ sudo ovs-ofctl add-flow s4 priority=500,in_port=1,dl_ty
pe=0x8000,nw_proto=6,actions=output:4
2022-10-12T00:55:49Z|00001|ofp_match|INFO|normalization changed ofp_match, deta
ils:
2022-10-12T00:55:49Z|00002|ofp_match|INFO| pre: in_port=1,dl_type=0x8000,nw_pro
to=6
2022-10-12T00:55:49Z|00003|ofp_match|INFO|post: in_port=1,dl_type=0x8000
yuji@yuji-VirtualBox:~$ sudo ovs-ofctl add-flow s3 priority=500,in_port=1,dl_ty
pe=0x8000,nw_proto=6,actions=output:1
2022-10-12T00:56:23Z|00001|ofp_match|INFO|normalization changed ofp_match, deta
ils:
2022-10-12T00:56:23Z|00002|ofp_match|INFO| pre: in_port=1,dl_type=0x8000,nw_pro
to=6
2022-10-12T00:56:23Z|00003|ofp_match|INFO|post: in_port=1,dl_type=0x8000
yuji@yuji-VirtualBox:~$ sudo ovs-ofctl add-flow s1 priority=500,in_port=1,dl_ty
pe=0x8000,nw_proto=6,actions=output:1
2022-10-12T00:57:04Z|00001|ofp_match|INFO|normalization changed ofp_match, deta
ils:
2022-10-12T00:57:04Z|00002|ofp_match|INFO| pre: in_port=1,dl_type=0x8000,nw_pro
to=6
2022-10-12T00:57:04Z|00003|ofp_match|INFO|post: in_port=1,dl_type=0x8000
```

Other H2 => H1

```
yuji@yuji-VirtualBox:~$ sudo ovs-ofctl add-flow s4 in_port=1,actions=output:1
yuji@yuji-VirtualBox:~$ sudo ovs-ofctl add-flow s2 in_port=1,actions=output:3
yuji@yuji-VirtualBox:~$ sudo ovs-ofctl add-flow s5 in_port=1,actions=output:3
yuji@yuji-VirtualBox:~$ sudo ovs-ofctl add-flow s3 in_port=1,actions=output:1
yuji@yuji-VirtualBox:~$ sudo ovs-ofctl add-flow s1 in_port=1,actions=output:1
```

(h) Also submit your custom topology files (.py files) for both topologies (part 2 & part 3) to Brightspace along with your report (do NOT paste code in report).

We have zero tolerance to forged or fabricated data!! A single piece of forged/fabricated data would bring the total score down to zero.